Cell-based approach for 3D reconstruction from incomplete silhouettes

Maarten Slembrouck[1], Peter Veelaert[1], David Van Hamme[1], Dimitri Van Cauwelaert[1], and Wilfried Philips[1]

imec - TELIN-IPI,
Ghent University, St-Pietersnieuwstraat 41, 9000 Gent, BELGIUM,
maarten.slembrouck@ugent.be,
WWW home page: http://telin.ugent.be/~mslembro

Abstract. Shape-from-silhouettes is a widely adopted approach to compute accurate 3D reconstructions of people or objects in a multi-camera environment. However, such algorithms are traditionally very sensitive to errors in the silhouettes due to imperfect foreground-background estimation or occluding objects appearing in front of the object of interest. We propose a novel algorithm that is able to still provide high quality reconstruction from incomplete silhouettes. At the core of the method is the partitioning of reconstruction space in cells, i.e. regions with uniform camera and silhouette coverage properties. A set of rules is proposed to iteratively add cells to the reconstruction based on their potential to explain discrepancies between silhouettes in different cameras. Experimental analysis shows significantly improved F1-scores over standard leave-M-out reconstruction techniques.

Keywords: shape-from-silhouettes, 3D reconstruction, occlusion, multi-camera

1 Introduction

Shape-from-silhouettes algorithms are very sensitive to errors in the provided silhouettes. Two types of errors are common: inaccurate silhouette boundaries and parts of the silhouette that are missing entirely. Such incomplete silhouettes may be due to errors in the segmentation algorithm, such as foreground/background segmentation, but their primary cause is occlusion. If a static object is positioned between the camera and the moving object, foreground/background segmentation is unable to segment parts of the silhouette. In indoor as well as outdoor environments, occlusion seems to be inevitable. Examples of occluding objects are furniture in an indoor setting or parked cars in outdoor setting.

Although in some circumstances, it may be possible to manually mark the occluding objects in camera images, for example, during an interview with fixed cameras, in most applications manual occlusion marking is impractical [2]. Alternatively, the presence of occluders has also been inferred from depth images, where depth information is provided either by stereo cameras [3][8][9][7] or by

depth cameras [1]. In a multi-camera system both options are expensive, since depth images are needed from each camera viewpoint. Furthermore, the detection of which parts are occluded in each view is only part of the problem: even with known occlusion a set of rules is required to reconstruct the shape as closely as possible.

The basic shape-from-silhouettes reconstruction has been around since 1994 when Laurentini introduced the visual hull [5], but applying it to incomplete silhouettes results in severely incomplete shapes.

A straight-forward algorithm to cope with incomplete silhouettes, was presented in 2008 by Landabaso et al. [4]. The algorithm reconstructs all regions which project within the silhouettes of $N - M$ cameras, where M is the number of incomplete silhouettes. The main difficulty, however, is that M is often unknown and can be different for different parts of the reconstruction. This algorithm often yields to larger reconstructions with ghost shapes enclosing the actual object.

The proposed algorithm in this paper is able to reconstruct a 3D shape geometrically from inconsistent silhouettes more accurately than the algorithm in [4]. Furthermore, no prior knowledge of the amount of occlusion is required. We partition the reconstruction space in different regions, called cells. These cells have uniform camera and silhouette properties. The aim is to find the union of cells, which cover the original 3D object. Therefore, we use geometric reasoning on the level of these cells. The algorithm is iterative and identifies the most likely cell to be added to the reconstruction. A stop criterion is also proposed that balances shape completeness and potential spurious additions.

In the remainder of this paper, we present our cell-based geometric reasoning framework and explain how this framework helps us to reconstruct a 3D shape from incomplete silhouettes. An example is given in Section 3. Section 4 shows an experiment where we simulate incomplete silhouettes in a camera setup and compare the results of the proposed reconstruction with a number of other approaches.

2 Cell-based geometric reasoning

The cells we define are continuous regions in space formed by the backprojected general cones of the silhouettes. The intersections of the cone boundaries delimit regions of 3D space with uniform camera segmentation information, e.g. all points within one such region fall inside the cones of one particular set of cameras and outside the cones of the silhouettes of the remaining cameras. We use geometric reasoning on cells, rather than on smaller entities, such as 3D points for two reasons. Each object usually consists of a 3D connected set of points (rather than a scattered set of points in the reconstruction space) and, given the input silhouettes and the camera calibration, the cells are the smallest coherent entities. In the next subsection we formally define the cells.

2.1 Space partitioning in cells

Let I_j be the silhouette of an object $\mathcal{S}$ with respect to viewpoint V_j. We define the projection of a point $P \in \mathbb{R}^3$ on the image sensor of camera j as P_j. For each point $P \in \mathbb{R}^3$ we define a membership function $\psi_j(P)$ as follows:

$$\psi_j(P) = \begin{cases} 1 & \text{if } P_j \in I_j \\ 0 & \text{otherwise,} \end{cases} \tag{1}$$

which indicates whether or not the projection of P lies inside or outside the silhouette of a particular camera. For N cameras we define the membership vector

$$\boldsymbol{\psi}(P) = (\psi_1(P), \ldots, \psi_n(P)). \tag{2}$$

That is, $\boldsymbol{\psi}(P)$ will be a binary vector of the form $(\ldots, 0, \ldots, 1, \ldots)$ that indicates for which cameras the projection of P is within the silhouette, and for which cameras it is not. When no occlusion is present the membership vector will be of the form $(1, 1, \ldots, 1)$ not only for the points in $\mathcal{S}$, but also for the points in the classical visual hull based on the silhouettes. The points outside the classical visual hull are the points P for which at least one element of $\boldsymbol{\psi}(P)$ is zero. When the silhouette is incomplete, however, some elements of $\boldsymbol{\psi}(P)$ may be zero even when P belongs to $\mathcal{S}$. Figure 1 shows a number of these membership vectors.

Now let P be any point in $\mathbb{R}^3$, then the cell A is defined as the set of all points $Q \in \mathbb{R}^3$ for which there exists a continuous path from Q to P such that $\boldsymbol{\psi}(Q) = \boldsymbol{\psi}(P)$ for all points along the path. Thus, a cell has the following properties:

1. all points in a cell share the same membership vector;
2. a cell is path-connected;
3. a cell is maximal in the sense that there are no points outside A for which there is a continuous path to a point inside the cell, such that all points along the path have the same membership vector.

Figure 1 shows a 2D example of the partitioning of the reconstruction space into cells using the 1D equivalent of the membership function. The colours of the cells represent the number of ones in the membership function, we call that the membership count of the cell.

Note that each point in space thus becomes part of a cell, and that at least one cell is infinitely large. Standard leave-M-out [4] adds cells based purely on membership count. In the next section, we will introduce other properties of cells which can be indicative of their likelihood of contributing to a more accurate 3D reconstruction.

2.2 Cell types

In the previous section, we defined how space is partitioned into cells based on the input silhouettes. In this section, we explain how the proposed algorithm is

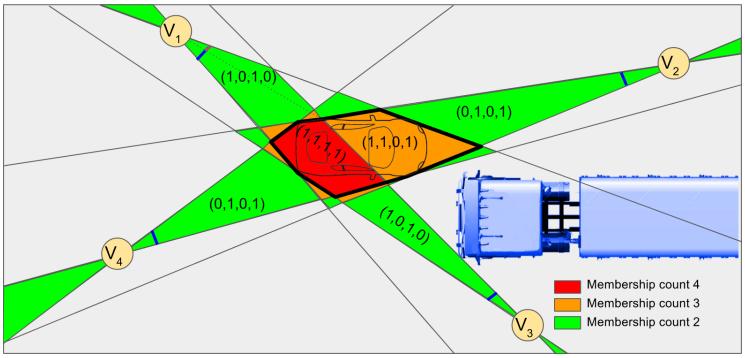

Fig. 1. Example of our algorithm in 2D with 4 viewpoints in case a stationary truck is partially blocking the view of camera 3. The aim is to find those cells which are part of the car. Different cells with membership count 2, 3 and 4 (number of ones in the membership function) are coloured as these are the cells which have to be evaluated. In some of the cells, we printed the cell's membership vector $\boldsymbol{\psi}$.

able to select a set of cells that well approximates the shape of the object of interest even in the presence of incomplete silhouettes.

To explain the algorithm, we will use the simple example in Figure 1. Although the example is in 2D, the geometric reasoning behind it, can immediately be extended to 3D objects. The aim is to reconstruct the car at the center, which camera 3 can only partly observe due to the truck blocking its view. The classical shape-from-silhouette algorithm will only reconstruct the red cell, because this cell is the only part of the car visible by all cameras. It is clear that an occlusion handling algorithm should also include the large orange cell in the reconstruction. Adding the large orange cell resolves the discrepancies between the reconstructed shape and the views of camera 1, 2 and 4. Clearly, this cell is a better candidate to be added to the reconstruction than the other cells with equal count of membership (all other orange cells). The next paragraphs formalize the method for identifying high-potential cells and adding them one by one.

The reconstruction starts from the cell(s) with membership count equal to the number of cameras, Y^0 to which cells A_i can be added one by one: $Y^{t+1} = Y^t \cup A_i$. This process continues until we have found a reconstructed object Y^n that explains the original silhouettes as well as possible. The choice of cell A_i is not obvious from membership count alone.

The problem of selecting the most appropriate cell at each step of the iteration t can only be resolved by a careful comparison of the cell silhouettes $I(A_i)_j$, the reconstructed object silhouettes $I(Y^t)_j$, and the original silhouettes I_j. To this end we will assign a type to each cell and for each view, which are based on these three silhouette types.

The silhouettes I_j and $I(Y^t)_j$ may either be disjoint, partially overlap, or one silhouette may be completely included into the other. Figure 2 illustrates the six different cases that are possible. The figure shows for a given camera view, the original 2D silhouette I_j and the projection of the current reconstruction $I(Y^t)_j$. Also shown are the projections of 6 different types of cells. These are the only possibilities that can occur. In particular, because of the way the cells have been defined, no cell can exist whose projection crosses the boundary of I_j.

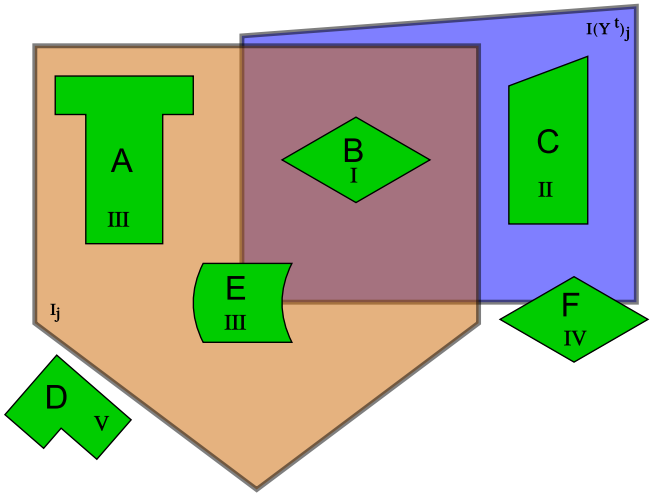

Fig. 2. A cell can take one of six possible cell positions relative to I_j and $I(Y^t)_j$, where $I(Y^t)_j$ denotes the silhouette of the current reconstruction Y^t. Since Y^t can grow, the cell type can change during the reconstruction process. For example, as soon as $I(Y^t)_j$ encompasses cell E, the cell's type will change from III to I.

Table 1 gives an overview of the 5 cell types, with a short description of the role a cell may play when added to the reconstruction Y^t. We briefly describe each cell type that appears in Figure 2.

A cell's type can be determined by a simple decision tree, as illustrated in Figure 3. When the reconstructed object Y^t gets larger, the cell types may change. Not all transitions are possible, however. For example, if a cell's projection is not part of I_j then its type can never change to I or III. Similarly, type IV may become type II, but type II cannot change to IV, since $I(Y^t)_j$ will never shrink. The dashed arrows in Figure 3 indicate the four transitions that are possible between the types.

From the standpoint of occlusion handling, cells A and E are the most interesting. Cells A and E are both of type III for this camera j since their projections are part of I_j, but the cells are not yet included in $I(Y^t)_j$. Either of these cells could be added to the reconstruction Y^t to explain a larger part of the original

type	description
I	unnecessary, but unharmful (part of input silhouette)
II	unnecessary, but unharmful (part of detected occluded area)
III	adding cell better explains I_j
IV	unnecessary, but may be part of previously detected occluder
V	unnecessary, and creates additional occluder

Table 1. Cell types and their meaning. Note that the cell type is specific to the considered camera.

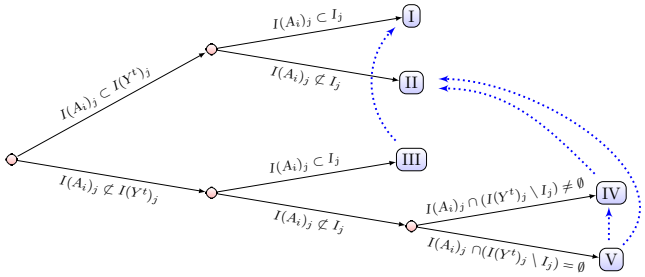

Fig. 3. The five different cell types for one camera view are shown in this decision tree. Since I_j is fixed and only $I(Y^t)_j$ can grow during the different steps in the algorithm, not all transitions are possible. The blue dashed lines show the possible transitions.

silhouette I_j. Cell B is of type I. Adding this cell is unnecessary to explain the original silhouette I_j. However, since it does not contradict silhouette I_j, it may be added without introducing new unexplained parts to $I(Y^t)_j$.

Cell C is of type II. Although the projection of cell C lies outside I_j, it lies inside $I(Y^t)_j$, which means that previous steps in the reconstruction have shown that cell C may belong to an occluded part of the object. Although adding cell C is unnecessary to explain I_j, it may do no harm. To a lesser extent the same remark holds for cell F, which is of type IV. Although its projection is not part of I_j, the projection overlaps partially with $I(Y^t)_j$ indicating that an occluder may block the view of cell F, since the visual cone of $I(Y^t)_j$ meets cell F. Finally, cell D, which is of type V, is the least likely to become part of the reconstruction. Although other cameras may need cell D to explain their silhouettes, for this particular camera view cell D is not necessary. Moreover, adding cell D assumes the presence of an occluder for which there was no other evidence so far.

This simple example suggests that, based on the membership vector, simple effective rules can be used to decide which cells should be added to the reconstruction. We will now further elaborate these rules.

2.3 Counting functions

As we noted previously, a cell's type may be different for each camera view. For example, cell A_i may be of type I for camera 1, but of type III for camera 2. Whether a cell will be added to the reconstruction should depend on how the cell is observed by each camera. To arrive at a ranking system, we introduce type counting functions $\chi_X(A_i, Y^t)$, one for each type, that count how many of the N cameras classify the cell as being of type X. The counting function not only depends on the cell A_i, but also on the current approximation Y^t, since Y^t will change during the reconstruction. For example, in a configuration of 4 cameras, if cell A_i is classified as being of type III, III, I, II, respectively, then $\chi_{\mathrm{I}}(A_i, Y^t) = 1$, $\chi_{\mathrm{II}}(A_i, Y^t) = 1$, $\chi_{\mathrm{III}}(A_i, Y^t) = 2$, $\chi_{\mathrm{IV}}(A_i, Y^t) = 0$, $\chi_{\mathrm{V}}(A_i, Y^t) = 0$.

Notice that the counting functions for each cell sum to N:

$$\chi_{\mathrm{I}}(A_i, Y^t) + \chi_{\mathrm{II}}(A_i, Y^t) + \chi_{\mathrm{III}}(A_i, Y^t) + \chi_{\mathrm{IV}}(A_i, Y^t) + \chi_{\mathrm{V}}(A_i, Y^t) = N. \quad (3)$$

2.4 Cell priority and reconstruction

The reconstruction we want to obtain is one that explains the silhouettes while assuming as little occlusion as possible.

The reconstruction is an iterative process. Initially the reconstruction starts from the cells with membership count equal to the number of cameras N, e.g. the classical visual hull Y^0. At each iteration step, a new cell A_i is added to Y^t. We now propose a simple mechanism to select the most appropriate cell A_i. The selection is based on the current values of the counting functions. More precisely, for each cell we define the *score vector*

$$\mathrm{W}(A_i, Y^t) = \left(\chi_{\mathrm{III}}(A_i, Y^t), \chi_{\mathrm{I}}(A_i, Y^t) + \chi_{\mathrm{II}}(A_i, Y^t), \chi_{\mathrm{IV}}(A_i, Y^t)\right), \quad (4)$$

whose elements are simple linear combinations of the counting functions.

After computing the vector $\mathrm{W}(A_i, Y^t)$ for all cells that are not in Y^t, the cells are sorted using a descending lexicographic order on their score vectors $\mathrm{W}(A_i, Y^t)$. In front of the list will be the cells that have the largest value for $\chi_{\mathrm{III}}(A_i, Y^t)$. These are cells that match the silhouette I_j of many cameras, but are not yet part of the current reconstruction Y^t. When two cells have the same value for $\chi_{\mathrm{III}}(A_i, Y^t)$ a further distinction is made based on the value of $\chi_{\mathrm{I}}(A_i, Y^t) + \chi_{\mathrm{II}}(A_i, Y^t)$. Thus, cells whose projection is already covered by the projection of Y^t for one or more camera views will have a larger priority in the list. Finally, when there is still a draw between cells, $\chi_{\mathrm{IV}}(A_i, Y^t)$ is used in order to give preference to cells whose projection is at least partially covered by $I(Y^t)_j$ in one of the cameras.

The reconstruction is now straightforward. After sorting the cells, the cell A_i that is in front of the list is added to Y^t, and the score vectors $\mathrm{W}(A_i, Y^{t+1})$ are recomputed for all cells that are not in Y^{t+1}. This process is repeated as long as there are cells for which $\chi_{\mathrm{III}}(A_i, Y^t) > 0$. Algorithm 1 shows the pseudocode.

The proposed approach is generic in the sense that the vector $\mathrm{W}(A_i, Y^t)$ can be replaced by any other vector whose elements are combinations of the

Data: Original silhouettes I_j
Result: reconstructed object (Y)
Set Y^0 equal to the union of cells with membership count N, $t = 0$
Use the silhouettes to construct the list of all cells Φ that are not in Y^0
while $\chi_{\text{III}}(A_i, Y^t) > 0$ *for at least one cell in the list* Φ **do**
    compute the score vectors $\text{W}(A_i, Y^t)$
    sort the cells using a descending lexicographic order on $\text{W}(A, Y^t)$
    select the cell A_i that comes first in the sorted list
    set $Y^{t+1} = Y^t \cup A_i$, remove A_i from the list Φ and increment t by one
end

Algorithm 1: Occlusion detection and handling algorithm.

counting functions. However, we found that the vector proposed in (4) yields a reconstruction which contains the entire object with the lowest number of cells.

Also note that during the reconstruction the values of both $\chi_{\text{III}}(A_i, Y^t)$ and $\chi_{\text{IV}}(A_i, Y^t)$ will gradually decrease, since some cells will change their type to I or II when Y^t grows larger. For the same reason, $\chi_{\text{I}}(A_i, Y^t)$ and $\chi_{\text{II}}(A_i, Y^t)$ will gradually increase. Hence, after adding the cells with a high value for $\chi_{\text{III}}(A_i, Y^t)$, (that is, the cells that explain many of the original silhouettes), the algorithm will gradually shift its attention to cells that may explain fewer of the original silhouettes, but that provide a match with the silhouettes of the reconstructed object.

3 Example

Figure 4 shows a more challenging example of a 2D reconstruction with 2 occluders and 4 cameras. The view of the car at the center is partially blocked by two static occluders (stationary trucks) in a setup with four cameras. Table 2 lists the different iterations of the algorithm. The second row in Figure 4 shows the order in which the cells are added to the reconstruction in each iteration step. We notice that the algorithm needs 5 iterations to complete the reconstruction. Note that cells with $\chi_{\text{III}} = 0$ are discarded in the next iteration because they are no longer of interest. The algorithm consequently adds cells Y^0, B, A and K.

4 Experiments

The choice of the score vector was based on a large set of experiments. For these experiments, we use the JP sequences (breakdancer) from the CVSSP-3D dataset [6]. Each of these sequences consists of a synchronised stream of images from 8 cameras, which are placed around the subject at 2.2m high about every 45 degrees (see Figure 6). A total of six sequences is available. In this experiment we will simulate the presence of occluding objects and compare with the ground truth, which is the output of the classical shape-from-silhouettes without occlusion (see Figure 5). Each frameset has equal weights on the evaluation.

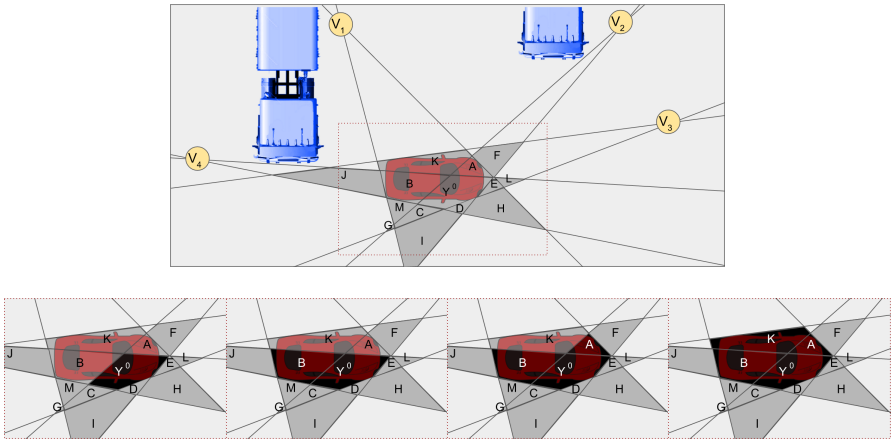

Fig. 4. Example to demonstrate how the algorithm works. There are two stationary trucks present, acting as two static occluders for cameras 2 and 4. Reconstruction during the different iterations based on the results in Table 2. From left to right: set of added cells (in black) after iteration 1 until 4.

it	cell	V_1	V_2	V_3	V_4	W(A, Y^t)	it	cell	V_1	V_2	V_3	V_4	W(A, Y^t)
1	**Y^0**	III	III	III	III	**(4, 0, 0)**	3	**A**	I	I	III	V	**(1, 2, 0)**
2	A	I	I	III	V	(1, 2, 0)		C	I	I	I	V	(0, 3, 0)
	B	III	V	III	I	**(2, 1, 0)**		F	V	I	III	V	(1, 1, 0)
	C	III	I	I	V	(1, 2, 0)		I	I	I	V	V	(0, 2, 0)
	D	I	I	V	I	(0, 3, 0)		J	V	IV	III	I	(1, 1, 1)
	E	I	V	I	I	(0, 3, 0)		K	I	IV	III	V	(1, 1, 1)
	F	V	I	III	V	(1, 1, 0)		M	I	II	I	V	(0, 3, 0)
	G	V	I	I	V	(0, 2, 0)	4	C	I	I	I	V	(0, 3, 0)
	H	I	V	V	I	(0, 2, 0)		F	V	I	III	IV	(1, 1, 1)
	I	III	I	V	V	(1, 1, 0)		J	V	IV	III	I	(1, 1, 1)
	J	V	V	III	I	(1, 1, 0)		**K**	I	IV	III	IV	**(1, 1, 2)**
	K	III	V	III	V	(2, 0, 0)	5	F	V	I	I	IV	(0, 2, 1)
	L	V	V	I	I	(0, 2, 0)		J	V	IV	I	I	(0, 2, 1)
	M	III	V	I	V	(1, 1, 0)							

Table 2. Example of the algorithm with two stationary trucks as occluders, captured by four cameras. Cells in bold are added to the reconstruction volume. Each iteration is based on the corresponding result from the previous iterations (Figure 4). Underlined class labels have changed compared to the previous iteration. From iteration 2 we list all candidate cells, which form the search space. Added cells and cells with $\chi_{\text{III}} = 0$ are discarded in the next iteration.

Displayed values are therefore averages over each sequence. Reconstructions are discretized at a regular voxel size of 20 mm.

We compare against 5 methods. The classical shape-from-silhouettes (CSfS [5]) is the standard method for geometric shape reconstruction, but has no occlu-

Fig. 5. Some examples from the reconstruction of the CVSSP-3D dataset [6]

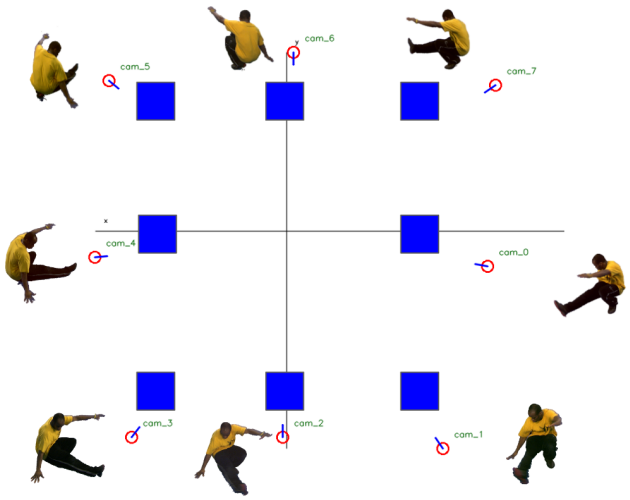

Fig. 6. Visualization of the occluders. Each camera has one possible occluder (blue) which can be either turned on or off. The eight cameras are facing the same area. All cameras are mounted at approximately 2.2m from the ground plane in a square configuration (on each vertex and in the middle of each edge).

sion handling. A second method is to ignore camera views with partial occlusion immediately and reconstruct the visual hull from the remaining camera views (IOC). Another method is to mark the occluded pixels in each of the images as foreground and perform classical shape-from-silhouettes with these adapted silhouettes [2], which is a tedious task and it defies the purpose of automatic occlusion detection. The fourth method is only usable in simulation because it uses the ground truth shape to determine the cells (produced in the same way as in our proposed algorithm) that are overlapping with the ground truth points. This method indicates how well a cell based method is able to perform. Finally, we also compare against the shape from inconsistent silhouettes method (SfIS [4]). Here we take the number M of accepted incomplete silhouettes equal to the number of occluded views. Note that all methods are naive. They either obtain high recall and low precision by overcompensating for occlusion (IOC, MOA, SfIS), or obtain high precision and low recall by not compensating for occlusion in any way (CSfS). Table 3 shows the results for all methods we compare to, aver-

aged over all sequences. Only the F1-score is shown. The last column represents the results of the proposed method. Figure 7 shows the results graphically.

#	CSfS	IOC	MOA	MESC	SfIS	proposed
0	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
1	66.60%	98.61%	99.31%	99.16%	94.77%	99.05%
2	47.56%	96.51%	98.36%	97.90%	87.20%	97.68%
3	36.89%	93.28%	97.01%	96.06%	77.40%	95.67%
4	30.36%	87.90%	94.92%	93.26%	62.82%	92.60%
5	25.79%	77.93%	91.18%	88.42%	43.53%	87.38%
6	22.35%	56.29%	82.34%	77.23%	19.98%	75.25%
7	19.65%	10.67%	56.54%	33.20%	3.57%	41.82%
8	17.45%	0.00%	17.47%	19.43%	0.74%	33.45%

Table 3. Results in case of multiple occluders. The first column indicates the number of occluders. The occluder is placed in front of a certain camera. Each occluder has the same dimensions (500x500x1500mm). The numbers indicate the average over all possible combinations.

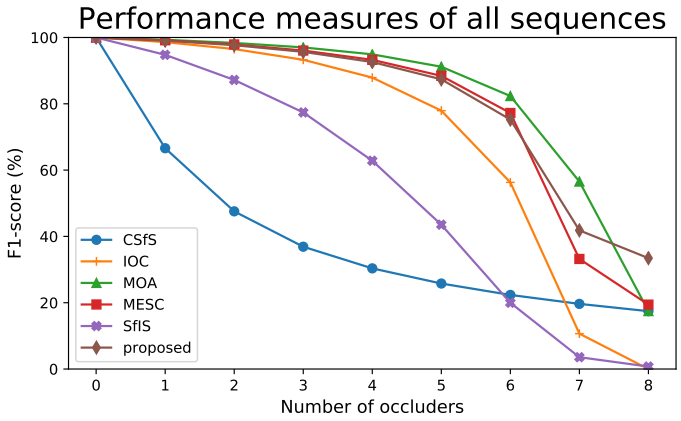

Fig. 7. Graphical representation of the values in table 3. We notice that our proposed method performs well compared to the other methods.

We notice that the result of the proposed method is close to the result of the minimal enclosing set of cells method (MESC) for up to 6 occluded views, which means that the rule set to rank cells is close to optimal. In the case of 8 occluders, there is no view left which detects the lower body parts of the person. The proposed method still produces meaningful results in this case, when the other methods perform badly. The reason is that the reconstructed volume does

not include the entire object, leading to a small number of false positives. As the F1-score gives equal weight to both recall and precision, our algorithm seems to perform better. However, since these are degenerate cases, we should look at the occlusion numbers which are more realistic (0-5). The main difference between the MOA results and the proposed method is the that adjacent cells sometimes are seen as one because of inaccurate camera calibration and errors on the edges of the silhouette detection.

5 Conclusion

In this paper we presented an algorithm for geometric shape reconstruction from incomplete silhouettes. The algorithm has two advantages over other shape-from-silhouettes algorithms. First, the proposed algorithm effectively handles occlusion and other errors in the silhouettes. Second, no manual or explicit occlusion detection is required, neither is any prior knowledge of the object being reconstructed.

Further research may focus on the applicability towards multi-object reconstruction and temporal tracking of occluders instead of frame-by-frame detections.

References

1. Ding, P., Song, Y.: Robust object tracking using color and depth images with a depth based occlusion handling and recovery. In: Fuzzy Systems and Knowledge Discovery (FSKD), 2015 12th International Conference on. pp. 930–935. IEEE (2015)
2. Guan, L., Sinha, S., Franco, J.S., Pollefeys, M.: Visual hull construction in the presence of partial occlusion. In: 3D Data Processing, Visualization, and Transmission, Third International Symposium on. pp. 413–420. IEEE (2006)
3. Kang, S.B., Szeliski, R., Chai, J.: Handling occlusions in dense multi-view stereo. In: Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on. vol. 1, pp. I–103. IEEE (2001)
4. Landabaso, J.L., Pardàs, M., Casas, J.R.: Shape from inconsistent silhouette. Computer Vision and Image Understanding 112(2), 210–224 (2008)
5. Laurentini, A.: The visual hull concept for silhouette-based image understanding. Pattern Analysis and Machine Intelligence, IEEE Transactions on 16(2), 150–162 (1994)
6. Starck, J., Hilton, A.: Surface capture for performance-based animation. Computer Graphics and Applications, IEEE 27(3), 21–31 (2007)
7. Sun, J., Li, Y., Kang, S.B., Shum, H.Y.: Symmetric stereo matching for occlusion handling. In: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05). vol. 2, pp. 399–406. IEEE (2005)
8. Wegner, K., Stankiewicz, O., Domański, M.: Occlusion handling in depth estimation from multiview video. In: Signals and Electronic Systems (ICSES), 2014 International Conference on. pp. 1–4. IEEE (2014)
9. Zitnick, C.L., Kanade, T.: A cooperative algorithm for stereo matching and occlusion detection. IEEE Transactions on pattern analysis and machine intelligence 22(7), 675–684 (2000)